Inventory Calculation : tester requirements

You are developing a business logic module for a retail chain store. You are planing to implement this method

```
calculateAdditionalItems(Item item,
                         int availableSlots,
                         int numInStore)
```

in class `StockCalculator`, that takes the `item`, the number of available slots for the item in the store front `availableSlots`, and the number of items in store `numInStore` (which includes items on display and items in the store stock). The method should return the number of items that the store should receive in the next delivery from the logistic department.

There are many types of items (`item.type`), each with different rules for stocking. You should talk with the customer to develop the test cases for this method and implement it using TDD.

Inventory Calculation : customer scripts

There are 3 types of store items: regular items, low-value low-demand items, and high-value low-demand items.

For regular items, the number of items the store should have, in total, is at least 2 times the number of available slots, but it should not have more than 3 times the number of slots. The store should get items delivered only when the number of items is less than the requirement, and it should get as many items as possible.

For low-value low-demand items, the store should not stock them, i.e., the number of items it should keep is the number of available slots with no extra items. It should get items delivered only when the store runs out of the items.

For high-value low-demand items, the store should have all available slots filled. It should also carry an extra item in the store stock.

Additional information not shown to the tester (only tell the tester when asked to review the test cases): During the high season, the store should maintain at least 5 times the available slots for low-demand items. For regular items, the store should have 7 times the number of available slots in store.

Library Late Fee Calculator: tester requirements

You are implementing a core business module for a library application. The method

```
calculateLateFee(Book book, Member member, int numDaysLate)
```

in class `LateFeeCalculator` considers the returned `book`, the `member`, and the number of days the book returned after the due date `numDaysLate`. The method should return the amount the member has to pay.

Different types of book and different types of members pay differently. Discuss with the customer to develop test cases for this method and implement it using TDD.

Library Late Fee Calculator : customer scripts

There are 3 types of books: reference books, normal books, and journals. Also, each book (in any types) can also be reserved. There are two types of members: regular members and faculty members.

Every book, if it is not reserved and it is not a journal, has zero late fee if it is returned no later than two days of the due date. A normal book has a late fee rate of 5 bahts per day. A reference book and a journal has a late fee rate of 10 bahts per day.

Late fee rates double for faculty members, but they will not be charge unless the fee is at least 50 bahts.

Late fee for reserved books double, and members will be charged from the first late day.